UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Chair of Practical Computer Science IV:
Dependable Systems Engineering

Master's Thesis

Vulnerabilities in Privacy-Preserving Record Linkage: The Threat of Dataset Extension Attacks

as part of the degree program Master of Science Business Informatics submitted by

Marcel Mildenberger
Matriculation number 1979905

on May 12, 2025.

Supervisor: Prof. Dr. Frederik Armknecht
PhD Student Jochen Schäfer

Abstract

The abstract should serve as an independent piece of information on your Thesis conveying a concise description of the main aspects and most important results. It should not be excessively long.

Write the abstract.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

Acronyms

AI Artifical Intelligence

ANN Artificial Neural Network

BF Bloom Filter

DEA Dataset Extension Attack

ELD Encoded Linkage Data

GMA Graph Matching Attack

ML Machine Learning

PII Personally Identifiable Information

PPRL Privacy-Preserving Record Linkage

PRNG Pseudo-Random Number Generator

SMC Secure Multi-Party Computation

TMH Tabulation MinHash

TSH Two-Step Hash

1. Introduction

Linking data and records is an important component of research, software development and software projects. The primary reason for integrating data from different sources is to gain richer, more comprehensive insights about the same entity. Initially, deterministic record linkage, which relies on exact matches between predefined identifiers such as unique IDs, was the main method used in early linkage techniques. However, deterministic approaches often fail in real-world scenarios where data may suffer from inconsistent formatting, typographical errors or missing values, making exact matches impossible [HSW07].

The introduction of a probabilistic framework for record linkage by Fellegi et al. in 1969 [FS69] marked a significant advance in overcoming these limitations. Their work, "A Theory for Record Linkage" [FS69], proposed a statistical method for linking records across datasets by calculating the probability that two records refer to the same entity, even when there are inconsistencies in the data. Their approach evaluates common attributes and assigns weights based on the likelihood of a match. By accounting for discrepancies in real-world data, the so-called Fellegi-Sunter model has become a fundamental methodology for data linkage, particularly in heterogeneous and distributed data environments where traditional deterministic methods fall short [FS69].

Such a probabilistic approach to record linkage is important in sectors such as healthcare and social sciences, where data is often distributed across multiple institutions or sources and lacks unique identifiers. In these fields, the ability to integrate datasets is essential for gaining insights and improving outcomes. In the United States, for example, the healthcare system is highly fragmented, consisting of numerous independent entities such as hospitals, clinics, insurance companies, public health agencies, and research institutions. Each of these organisations collects and stores patient data independently, often using different systems and, more importantly, different formats. This fragmentation creates significant challenges when trying to track patient outcomes, monitor disease outbreaks, or evaluate the effectiveness of treatments across populations. Effective data linkage can bridge these gaps by linking records that refer to the same individual across multiple datasets. This integration is critical for tasks such as epidemiological research, public health surveillance, personalised medicine and healthcare quality improvement [PSZ+24; VSCR17].

For example, during the COVID-19 pandemic, the inability to efficiently link data between testing centres, hospitals and vaccination sites limited timely tracking of infection rates and vaccination outcomes. Had more robust data linkage mechanisms been in place, public health officials could have responded more effectively to outbreaks and targeted interventions to specific populations. Data linkage thus plays a critical role in transforming fragmented data landscapes into unified and actionable insights, leading to more informed decision-making. In response to the COVID-19 pandemic, organisations such as the Centers for Disease Control and Prevention and the Food and Drug Administration have launched projects to address these challenges and further develop linkage techniques [PSZ+24].

In scenarios such as the COVID-19 pandemic, data integration efforts often involve linking records on natural persons from multiple sources. For example, integrating data from differ-

ent healthcare providers, laboratories and public health agencies typically requires the use of pseudo-identifiers derived from Personally Identifiable Information (PII), such as names, dates of birth or other sensitive information. However, reliance on PII for linkage raises significant privacy concerns, as improper handling of such data can lead to re-identification of individuals, with potentially serious consequences such as data breaches, identity theft or unauthorised access to personal health information [PSZ+24; SBR09].

The increasing digitisation of personal data has already led to large-scale data breaches, demonstrating the risks of improperly secured data. Notable incidents such as the Cambridge Analytica scandal, in which personal data was misused for political profiling, highlight the ethical and regulatory challenges of data integration [IH18]. Similarly, healthcare data leaks have raised concerns about the implications of unauthorised access to medical histories, genetic data and insurance records. Leaks of personal health information can have serious consequences, including blackmail, discrimination, and fraud, which can cause significant personal harm. For example, individuals whose medical histories are exposed may face discrimination in employment or insurance, while others may become targets of scams that exploit their health conditions. The potential for such abuse underlines the critical importance of robust data protection measures by working with PII [Smi16].

To address these privacy risks, various techniques have been developed to protect PII during the linking process, primarily by encoding the data prior to linking. However, the use of encoded PII as pseudo-identifiers presents additional challenges. The key question is how to efficiently decode sensitive information while maintaining the ability to accurately match records [SBR09].

Therefore, Privacy-Preserving Record Linkage (PPRL) techniques are designed to facilitate data integration without exposing sensitive information, ensuring that datasets can be securely linked across different entities. To enable linkage while preserving privacy, similarity preserving encoding is applied to the PII. Without such similarity preserving encoding, matches between encoded entities in different databases would not be possible [SBR09; VSCR17]. Over time, three main privacy-preserving encoding schemes have emerged as enablers for PPRL [SAH24; VCRS20].

Bloom Filter (BF) encoding is the most widely used technique in PPRL and is often considered the reference standard [SAH24]. Originally introduced by Burton Bloom in 1970 as a probabilistic data structure for efficient set membership testing [Blo70], BFs were later adapted for PPRL due to their simplicity and efficiency in both storing and computing set similarities. Their compact representation and probabilistic nature make them ideal for scalable PPRL systems, especially in environments dealing with large datasets [SBR09]. The seminal work of Schnell et al. [SBR09] demonstrated the use of BFs in PPRL, particularly in healthcare, highlighting their ability to perform secure record matching without exposing sensitive identifiers [SBR09].

However, BFs are not without limitations. Their vulnerability to graph-based attacks and pattern exploitation has driven research into improving their security. Techniques such as diffusion have been proposed to obscure recognisable patterns and increase security [AHS23; SAH24]. For example, Armknecht et al. [AHS23] explored methods to strengthen the security of BF by adding a linear diffusion layer to the BF-based PPRL approach, which complicates pattern mining attacks.

To address some of these weaknesses of BFs, Tabulation MinHash (TMH) encoding has been introduced as a more secure alternative. MinHash, first developed by Broder in 1997 for estimating set similarities in large document collections [Bro97], has been adapted using

tabulation-based hashing [Smi17]. Although less widely used than BFs, TMH offers distinct advantages, including stronger security guarantees against re-identification attacks. However, these benefits come at the cost of increased computational complexity and memory usage, which may limit its applicability in resource-constrained environments [Smi17].

A further development in encoding techniques is the introduction of Two-Step Hash (TSH) encoding, which aims to combine the strengths of both BFs and TMH while mitigating their respective weaknesses. As detailed by [RCS20], TSH employs a two-stage process: data is first encoded using multiple BFs, followed by an additional hashing layer that transforms the encoded data into a set of integers suitable for similarity comparison. This layered approach enhances privacy by adding an extra layer of obfuscation, making it more resistant to attack, while maintaining efficient similarity computations [RCS20; VCRS20].

In practice, BF-based PPRL has become the dominant standard and is widely used in areas such as crime detection, fraud prevention and national security due to its balance of efficiency and ease of implementation. However, BF-based PPRL systems are not without limitations and vulnerabilities. Previous research has shown that there are several attacks targeting PPRL systems, with a focus on exploiting the weaknesses inherent in BF encodings. These attacks specifically target weaknesses in BF constructions, such as the weaknesses introduced by possible double hashing, structural flaws in filter design, and susceptibility to common pattern-mining techniques. Notably, no specific attacks have been developed for TMH or TSH encodings, suggesting that research has focused primarily on the more widely used BF scheme [VCRS20].

However, a more recent and practical attack has emerged that exploits vulnerabilities common to all PPRL encoding schemes. The Graph Matching Attack (GMA) uses publicly available data, such as telephone directories, to re-identify encoded individuals based on overlapping records between plaintext and encoded databases [SAH24; VCRS20]. Unlike previous attacks that focus solely on the encoding scheme of BFs, the GMA works independently of the encoding scheme chosen. It therefore exploits the graph structure of encoded datasets to re-identify records. Given two datasets - a plaintext reference dataset and an encoded dataset - an attacker can construct similarity graphs where nodes represent individuals and edges represent similarity scores. By solving a graph isomorphism problem, attackers can infer one-to-one mappings between encoded and plaintext records, effectively breaking the privacy guarantees of PPRL. The effectiveness of GMAs depends on the overlap between the two sets of data; the greater the overlap, the higher the probability of successful re-identification. While GMAs can successfully re-identify individuals present in both the plaintext and encoded datasets, their effectiveness is limited to the overlapping subset of the two databases [SAH24; VCRS20].

This work aims to go beyond traditional GMAs by re-identifying not only individuals present in the overlapping datasets, but as many individuals as possible from the encoded PPRL data. To achieve this, the newly introduced Dataset Extension Attack (DEA) builds on the foundations laid by GMAs. The DEA uses an Artificial Neural Network (ANN) trained on the subset of previously re-identified individuals to predict and decode the remaining encoded records. In doing so, the DEA significantly expands the scope and effectiveness of the attack, enabling broader de-anonymisation of PPRL datasets beyond the limitations of existing graph-based methods.

1.1. Motivation

The increasing use of PPRL in highly sensitive areas such as healthcare, finance and national security requires research to validate existing techniques and ensure robust privacy [SBR09]. As data-driven applications continue to evolve, the complexity and volume of data being collected and linked across multiple sources is growing rapidly. While PPRL systems are designed to facilitate secure data integration without compromising privacy, evolving cybersecurity threats and attack techniques highlight the urgent need to reassess the resilience of these systems [VSCR17].

Privacy has always been a critical concern in data management, but its importance has been intensified in the era of Artifical Intelligence (AI) and Machine Learning (ML). These technologies increasingly rely on large data sets, often containing sensitive PII such as medical records, financial transactions or behavioural data for training. If compromised, the exposure of such data can lead to serious privacy violations, including identity theft, financial fraud and discrimination. The rise of data brokerage, where personal information is collected, aggregated and sold - often without explicit user consent - further exacerbates privacy concerns. This commoditisation of personal data has made PII an attractive target for malicious actors, increasing the risk of unauthorised data linking and re-identification attacks. As AI models become more advanced, the demand for rich, high-quality data continues to grow, making privacy an increasingly pressing issue [KM24; MK19].

In this context, the vulnerability of PPRL systems to emerging attack methods is of particular concern. While PPRL techniques such as BF are designed to hide sensitive identifiers during the data linkage process, recent research has shown that these systems are vulnerable to GMAs. GMAs exploit the similarity preserving properties of common encoding schemes to re-identify individuals by comparing patterns in encoded records with those in publicly available plaintext records. This approach undermines the fundamental goal of PPRL: to protect sensitive data during the record linkage process. Although current GMAs are limited to re-identifying individuals present in both the encoded and plaintext datasets, even partial data exposure in highly sensitive areas can have serious consequences [SAH24; VCRS20].

The introduction of DEAs poses an even greater threat to the integrity of PPRL systems. Unlike GMAs, DEAs aim to extend the scope of re-identification to as many individuals as possible within the encoded database. Using ANNs trained on previously decoded data from GMAs, DEAs can predict and decode additional records, potentially leading to the complete de-anonymisation of entire encoded datasets. This represents a paradigm shift, as it challenges the viability of widely used PPRL techniques, such as BF-based encoding, which have been considered secure.

The primary motivation for this research is to proactively investigate and demonstrate the consequences of such advanced attacks in order to prevent their realisation in real-world scenarios. By exposing the potential vulnerabilities of PPRL systems, this work aims to demonstrate how attackers could exploit decrypted data to compromise privacy on a large scale. A successful implementation of the DEA will provide empirical evidence that state-of-the-art methods are insufficiently secure, highlighting the urgent need for more robust privacy-preserving techniques.

Furthermore, there is a notable gap in current research regarding the extension of attack capabilities beyond the intersection of datasets. While significant efforts have been made to address the vulnerabilities exposed by GMAs, there is a lack of comprehensive studies exploring how ML can be used to generalise these attacks and compromise entire databases. This research

aims to fill this gap by developing and evaluating the DEA, thereby contributing to a broader understanding of PPRL vulnerabilities.

By addressing this gap, this thesis aims to contribute to the body of knowledge on PPRL vulnerabilities and serve as a foundation for future research aimed at strengthening these systems. The knowledge gained from this study will not only enable the development of more secure PPRL techniques, but will also influence best practices in privacy and security.

1.2. Related Work

The study by Vidanage et al. [VCRS20] represents a significant advance in the field of PPRL through the introduction of a new attack method known as GMA. Their work begins with a comprehensive overview of PPRL systems and the similarity preserving encoding techniques commonly used, such as BFs. The GMA exploits weaknesses in these encoding schemes by exploiting their ability to preserve partial similarity information even after encoding. By constructing similarity graphs from both encoded and plaintext datasets, the GMA solves a graph isomorphism problem to align nodes and successfully re-identify individuals in the encoded dataset using publicly available sources such as telephone directories. This method demonstrates the universal applicability of GMAs across different PPRL schemes, and highlights a critical weakness in systems previously thought to be more robust [VCRS20].

Building on this foundation, Schäfer et al. [SAH24] revisited and extended the work of Vidanage et al. Their contribution lies in a meticulous reproduction and replication of the original GMA, during which they identified a critical flaw: an undocumented pre-processing step in the provided codebase that inadvertently increased the effectiveness of the attack. While this step was originally intended to improve computational performance, it introduced errors into the proposed GMA. Schäfer et al. corrected this problem and further optimised the GMA, resulting in improved robustness and efficiency. Their improved implementation achieved higher re-identification rates compared to the original approach. This improvement not only validates the vulnerabilities highlighted by the GMA, but also highlights the potential for refining attack methodologies to expose even greater weaknesses in PPRL systems.

The work of Schäfer et al. [SAH24] is particularly relevant to this thesis, as their improved GMA implementation and accompanying codebase form the basis of the DEA proposed in this study. While the GMA is limited to re-identifying individuals present in both encoded and plaintext datasets, the DEA seeks to extend the scope of re-identification beyond this intersection. Using ANNs trained on the re-identified individuals from the GMA, the DEA aims to predict and decode additional datasets, potentially leading to complete de-anonymisation of encoded datasets.

To date, no existing research has proposed an approach comparable to the DEA. This thesis addresses this gap by developing and evaluating the DEA, thereby contributing to a broader understanding of PPRL vulnerabilities and highlighting the urgent need for more secure data linking techniques.

1.3. Contribution

The contribution of this thesis is divided into three main parts. First, a comprehensive analysis of PPRL systems is carried out, with particular emphasis on the three main encoding schemes: BF encoding, TSH encoding and TMH encoding. This analysis aims to highlight the

basic principles, strengths and weaknesses of each encoding scheme, setting the stage for the subsequent investigation of their susceptibility to DEA.

Next, the current state of the art GMA is analysed and its limitations are discussed in detail. Although GMAs have proven effective in re-identifying individuals within overlapping datasets, their applicability is limited to the intersection of plaintext and encoded records. This inherent limitation highlights the need for more advanced attack strategies that can go beyond this.

The main focus of this thesis is the implementation and evaluation of the DEA, which attempts to outperform the capabilities of GMAs by decrypting a larger fraction of encoded records. To achieve this, the thesis examines the conceptual foundations, theoretical underpinnings and technical requirements of the DEA. Building on the initial re-identifications made by the GMA, the DEA employs a supervised machine learning approach, specifically using ANNs trained on previously decoded data to predict and re-identify remaining encoded records. This method significantly extends the scope of de-anonymisation in PPRL systems and provides a novel approach to current research.

The DEA is then evaluated against the three main PPRL encoding schemes. While the specific encoding scheme has minimal impact on the GMA, which is primarily based on solving a graph isomorphism problem, it plays a role in the DEA. This is due to the fact that the ANN has to be trained separately for each encoding scheme to account for the unique structural features and nuances of the encoding. However, the DEA is designed with adaptability in mind, ensuring that it can be effectively applied across different encoding schemes, thus increasing its generalisability and practical relevance.

Through this research, the thesis aims to answer critical questions about the robustness of PPRL systems. It investigates how effective supervised machine learning-based DEAs are at re-identifying the remaining entries that GMAs cannot decode. It also examines how different encoding schemes affect the performance and accuracy of the DEA, providing insight into which schemes are more susceptible to such attacks and why. By addressing these issues, the thesis contributes to a deeper understanding of the vulnerabilities inherent in PPRL systems and lays the groundwork for the development of more secure privacy preserving techniques.

1.4. Organization of this Thesis

This thesis is divided into four main sections: technical background, methodology, results, and conclusion.

First, an overview of PPRL systems is given, with particular emphasis on a thorough analysis of the most commonly used encoding techniques. Next, the existing GMA is introduced and explained in order to provide the basis for the study. In addition, an overview of ANNs is given to provide the necessary background knowledge.

Next, a detailed description of the attack model for the DEA is outlined, including how ANNs are used to enhance the attack. This is followed by an explanation of the actual implementation of the DEA, along with a discussion of the experiments conducted. The results of the DEA on different encoding schemes are then analysed and evaluated. Finally, the thesis concludes with a summary of the main contributions, a discussion of the broader implications, and suggestions for future research.

2. Background

This chapter provides an overview of the key concepts relevant to this thesis, including PPRL, various encoding techniques used in secure linkage and attacks. PPRL enables different organizations to link records belonging to the same individual across datasets while preserving privacy, making it a crucial tool. However, the security of such methods can depend on the encoding techniques used to transform sensitive data into a more privacy-preserving format [VCRS20].

To understand the vulnerabilities of PPRL systems, we examine three key encoding techniques: BF, TMH, and TSH, each offering different trade-offs between efficiency, privacy, and robustness against attacks. While these methods aim to prevent direct access to plaintext identifiers, they remain susceptible to adversarial techniques designed to infer or reconstruct the original data [SAH24; VCRS20].

One such adversarial approach is the GMA, which leverages structural similarities between encoded and non-encoded datasets to re-identify individuals. Although GMAs are good in reconstructing intersections of datasets, they are limited to the intersection of the two datasets. To extend GMAs beyond known intersections, we explore ANN, which can learn complex patterns in encoded data and increase identification rates [SAH24; VCRS20].

By integrating these concepts, this chapter establishes the necessary foundation for understanding the DEA introduced later in this thesis, demonstrating how machine learning techniques can be employed to bypass existing privacy-preserving mechanisms.

2.1. Overview of PPRL

PPRL enables the linkage of records from different databases that refer to the same individual while tying to preserve privacy. Traditional record linkage relies on unique identifiers, but these are often unavailable or inconsistent due to variations in formatting, spelling or missing data in a distributed environment. Linking records directly on plaintext data poses significant privacy risks. To mitigate these risks and prevent further threats to data security, regulations such as the European Union's General Data Protection Regulation have been established to govern the handling of PII [SAH24; VCRS20].

To enable record linkage using PII while trying to preserve privacy, PPRL employs similarity-preserving encoding on quasi-identifiers, allowing linkage to be performed on encoded representations rather than raw data. This approach protects identities while still facilitating record matching based on encoded similarities and probabilistic approaches [SAH24; VCRS20].

The security of PPRL schemes depends on the encoding techniques used. Broadly, PPRL methods fall into two categories: perturbation based techniques and Secure Multi-Party Computation (SMC) based techniques. SMC-based techniques provide strong security guarantees and high accuracy but suffer from computational and communication overheads. Conversely, perturbation-based techniques balance linkage quality, scalability, and privacy protection, making them more practical for real-world applications [VCRS20].

As seen in Figure 2.1 a typical PPRL system involves three main parties working together in four steps to enable the linkage while maintaining privacy. First data owners, who are responsible for maintaining their respective databases, D and D', encode the quasi-identifiers by agreeing on an encoding scheme with the corresponding parameters before then as a second step sharing their respective data sets with the linkage unit. The linkage unit, a trusted entity, performs the actual record linkage using only the encoded representations, without access to the original identifiers. Once the linkage is completed, the linkage unit assigns unique pseudonyms to the successfully linked records and returns the pseudonymized dataset as a third step to the data owners. The data owners then replace the linkage data with these pseudonyms before sending the dataset in the fourth and final step to the receipient requesting the data. In the case of a research project this could be an data analyst. The data analyst can merge the records based on the identifiers and proceed with further research and analysis, all while minimizing the risk of re-identification and protecting individual privacy [SAH24].

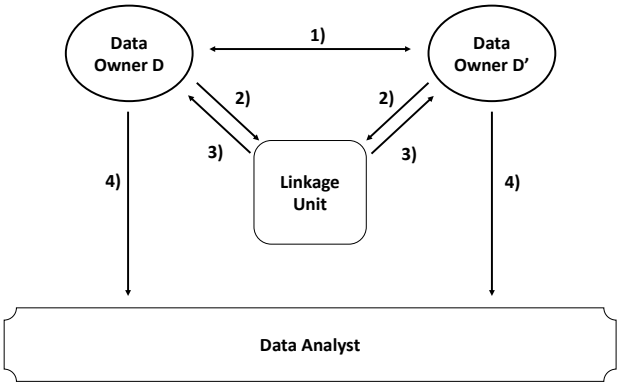

Figure 2.1.: Overview of the PPRL process.

Based on this scheme, each database record can therefore during the linkage process be represented as $r = (\lambda, \sigma)$, where λ denotes the linkage data which are encoded quasi-identifiers such as names and birthdates and σ refers to the remaining microdata like in a health care scenario patient information [SAH24].

Linkage in PPRL is performed probabilistically, meaning that two records, r and r', are considered linked if their similarity score on $sim(\lambda_r, \lambda_{r'})$ exceeds a predefined threshold. The choice of threshold plays a crucial role in balancing the quality of the linkage. Lower thresholds tolerate more variation and matches between records but increase the likelihood of false positives, while higher thresholds reduce false positives but may miss legitimate matches. Thus, selecting the optimal threshold is a trade-off that requires careful consideration of the specific goals and constraints of the linkage process [SAH24].

A robust PPRL scheme must satisfy several important criteria to ensure effective and secure linkage. First, a similarity function, $sim(\lambda_r, \lambda_{r'})$, must exist to determine if two records belong to the same entity based on a predefined threshold. Second, an encoding scheme $enc(\lambda)$ must be applied to λ in such a way that the linkage unit cannot reconstruct the original data. Finally, a function $sim(enc(\lambda_r), enc(\lambda_{r'}))$ must be available that allows similarity computations on encoded data which preserved similarty properties. These requirements ensure both the privacy

and the effectiveness of the record linkage process [SAH24].

One common approach for measuring similarity for two quasi identifiers and enabling probabilistic matching on quasi identifieres is using n-grams. Here, string values are divided into overlapping substrings of length n using a sliding window approach [SAH24]. For example, for the string "encoding" with $n = 2$, the n-grams are:

$$\{\text{en}, \text{nc}, \text{co}, \text{od}, \text{di}, \text{in}, \text{ng}\}$$

The similarity of two sets of n-grams can then be computed using metrics such as the Dice Coefficient where X and Y denotes two sets which are in the case of PPRL the n-grams for two different pseudo identifiers [SAH24].

$$Dice(X, Y) = \frac{2 \times |X \cap Y|}{|X| + |Y|}$$

The Jaccard Similarity can be used in a similar way [SAH24].

$$Jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

PPRL has been successfully applied in various domains, demonstrating its practical importance in securely linking records across institutions while assumingly preserving privacy. Notable applications include the Social Investment Data Resources, the Lumos Initiative, the Swiss National Cohort, and the Gemeinsamer Bundesausschuss. These implementations highlight the versatility of PPRL in diverse contexts, where privacy protection is essential while enabling effective data linkage for research and analysis [SAH24].

2.2. Key Encoding Techniques

In the context of PPRL, three primary encoding techniques have emerged: BF, TMH and TSH. These encoding methods are essential for transforming sets of quasi-identifiers into representations that preserve similarity information while supposingly maintaining privacy [SAH24; SBR09; VCRS20].

PPRL typically involves encoding sets of quasi-identifiers before linkage. A set in this context is generally a collection of n-grams, which are substrings of length n extracted from one or multiple attributes using a sliding window approach. Since input data varies in length, all encoding techniques in PPRL must take arbitrarily long inputs (sets of n-grams) and produce encoded outputs. This ensures that similarity computations can be efficiently performed on encoded data by the linkage unit without accessing the raw identifiers [SAH24; VCRS20].

Before linkage, data owners must agree on the encoding scheme and share necessary cryptographic secrets to facilitate secure comparison. Among the available techniques, BFs are the most widely used approach for record linkage [SAH24].

2.2.1. BF

BFs were originally developed for efficient membership testing in set structures without requiring direct access to the sets themselves [Blo70]. Due to their ability to efficiently compute set similarities in a privacy-preserving manner, they have been widely adopted in PPRL applications [SAH24; SBR09; VCRS20].

A BF $b \in \{0,1\}^l$ is a bit vector of length l. It uses $k \geq 1$ independent hash functions $H = \{h_1, h_2, ..., h_k\}$, where each function maps an arbitrary input to a position in the filter [SAH24; SBR09]:

$$h_i : \{0,1\}^* \rightarrow \{1, ..., l\}, \quad \forall i \in \{1, ..., k\} \tag{2.1}$$

Initially, the BF is set to all zeros. Each element $s \in S$ is hashed with every function h_i, and the corresponding bit positions in the filter are set to 1 [SAH24; SBR09]:

$$\forall s \in S, \forall h_i \in H, \quad b[h_i(s)] = 1 \tag{2.2}$$

An Example for this can be seen in Figure 2.2 where the set of 2-grams for the word "encoding" is encoded using a BF with $k = 2$ hash functions. As can be seen, the 2-grams are hashed using the two hash functions and the corresponding bits are set to 1 in the BF.

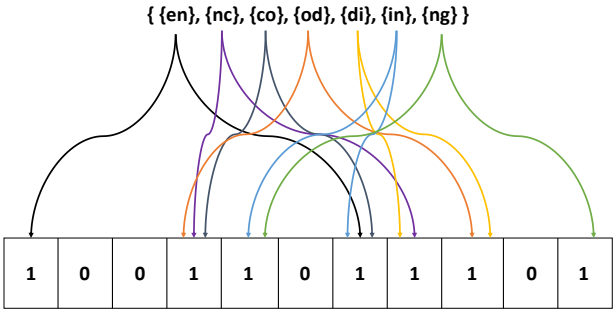

Figure 2.2.: Example BF for $k = 2$ hash functions on the set of 2-grams for "encoding".

Since BFs are binary vectors, the similarity between two BFs, b_1 and b_2, is computed based on the overlapping 1-bits based on their position. The Dice Coefficient is commonly used for this purpose [SAH24], providing a measure of similarity by comparing the overlap of 1-bits in the two binary vectors. Therefore encodings using BF encoding allows for efficient computation of similarity between two sets which is benefical for PPRL systems.

$$\text{Dice}(b_1, b_2) = \frac{2 \cdot |b_1 \cap b_2|}{|b_1| + |b_2|} \tag{2.3}$$

Because the sets in a PPRL systems consists of n-grams, a deterministic relationship between the n-grams present in the quasi-identifier λ and the set bits in the BF is created. However, due to the finite length of BFs, collisions occur where different n-grams map to the same bit position (see Figure 2.2). While this can cause incorrect linkages, it also enhances privacy by distorting frequency distributions [SAH24; VCRS20].

Three primary approaches exist for applying BFs to sensitive data. The first approach, Attribute-Level BFs, encodes each attribute, such as first name or last name, into a separate BF, enabling multiple similarity computations. However, Attribute-Level BFs are more vulnerable to frequency-based privacy attacks as they lower the collisions which would occur using only one BF. The second approach, Cryptographic Long-Term Key Encoding, merges

multiple attributes into a single BF, reducing vulnerability to frequency attacks but remaining susceptible to pattern-mining-based attacks. Finally, Record-Level BFs employ a weighted bit sampling technique to minimize frequency information, enhancing privacy protection while maintaining high linkage quality. Each of these approaches balances privacy concerns with the need for accurate and effective record linkage [VCRS20].

Several privacy-enhancing methods have been proposed to mitigate frequency attacks in BFs. These techniques introduce a trade-off between privacy and linkage quality. One such method is balancing, which ensures an equal number of 1-bits across BFs, thereby reducing the likelihood of frequency-based attacks. Another approach is salting, which randomizes bit positions to prevent direct inference from the encoded quasi-identifiers. Additionally, XOR folding is used to reduce the BF length while maintaining the bit-wise dependencies necessary for effective linkage. These methods aim to strengthen privacy while retaining the accuracy of the linkage process [SAH24; VCRS20].

A major improvement was introduced by Armknecht et al. [AHS23], who proposed a diffusion layer for BF encodings. This method generates Encoded Linkage Data (ELD), where each bit is computed as the XOR sum of multiple BF bits. The indices for XOR computations are randomly chosen and secretly shared among data owners [AHS23].

By applying diffusion, the deterministic relationship between 1-bits in the ELD and the original n-grams is broken, improving privacy while still enabling approximate matching [AHS23].

BFs remain a core technique in PPRL due to their efficiency and scalability. However, their vulnerability to frequency attacks has led to improvements such as Record-Level BFs and diffusion layers, which enhance privacy at the cost of increased computational complexity [AHS23; SAH24; VCRS20].

2.2.2. TMH

TMH is a variation of MinHash initially introduced for efficient estimation of set similarities and later adapted for privacy-preserving probabilistic record linkage. MinHash itself was first proposed in the context of document resemblance and containment estimation. TMH extends MinHash by employing tabulation-based hashing, which enhances its security compared to BFs [Bro97; VCRS20].

MinHash aims to approximate the Jaccard similarity between two sets, S and S'. The fundamental idea is to represent both sets as sequences of randomly ordered elements and apply multiple rounds of random permutations π to shuffle them. After each round, the first elements of both sequences are compared. The larger the intersection between S and S', the higher the probability that the first elements will match. An example for this can be seen in figure 2.3 where the sets are permutated twice and the first element is compared for each new set. The final Jaccard similarity estimate is computed based on the number of hash collisions achieved during permutation, in the example is one collision for two permutations [Bro97; SAH24; VCRS20].

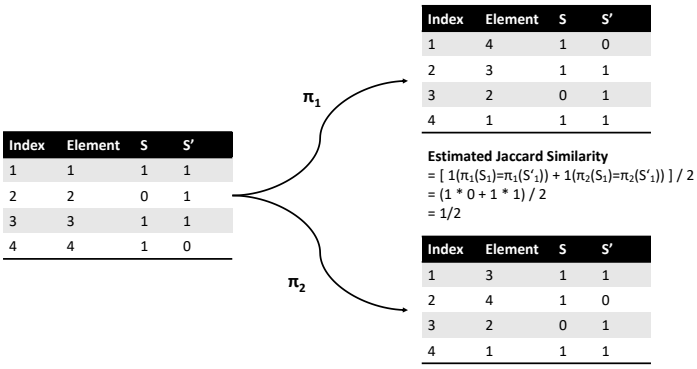

Figure 2.3.: Example computing approximate Jaccard similarity using MinHash with $\pi = 2$ permutations.

Instead of explicitly computing these permutations, MinHash simulates them by applying a suitable hash function to the elements of a set and selecting the smallest hash value as the representative signature. This is equivalent to sorting set elements by their hash values and returning the first element [SAH24].

Tabulation-based hashing is a technique used in TMH that provides efficient and high-quality hash functions by leveraging precomputed lookup tables. This method operates as follows [VCRS20]:

The process begins with the **initialization** of l sets of lookup tables, each containing c tables. Each table holds randomly generated bit strings for keys of length k, with a key space of 2^k. During the **hashing process**, each element in S is hashed using a one-way hash function, producing a fixed-length binary value. This binary value is then split into c sub-keys, each of length k. Each sub-key is used as an index to retrieve a random bit string from the corresponding lookup table, and the retrieved c bit strings are XORed together to produce a single output bit string. In the **MinHash signature generation** phase, this process is repeated for each of the l lookup table sets, and the minimum value among all generated bit strings is selected as the MinHash signature [SAH24; VCRS20].

An example for this can be seen in Figure 2.4 where the first element of the set is hashed using the first lookup table and the fourth hash function. The key is split into $c = 4$ sub-keys of length $k = 2$ and used to retrieve the corresponding bit strings from the lookup table. The retrieved bit strings are then XORed together to produce the final value for the first element.

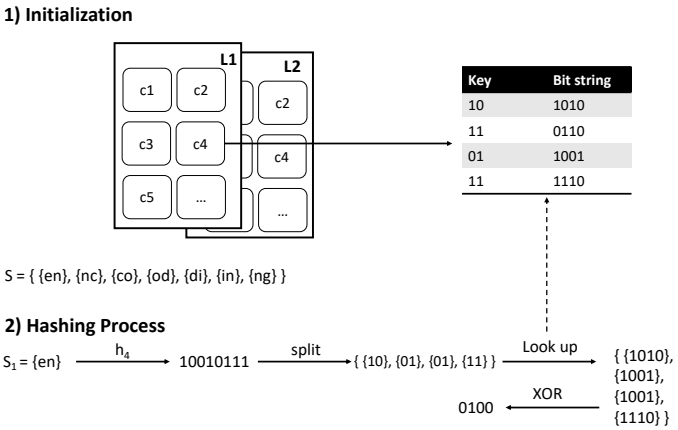

Figure 2.4.: Simplified TMH hasing step for the first lookup table, fourth hash function on the first set element.

To further enhance privacy, TMH employs a 1-bit hashing mechanism, where only the least significant bit of each MinHash signature is retained. These l bits are then concatenated to form the final bit array used as an encoded representation [SAH24].

The main advantage of TMH over BFs is its improved resistance to frequency-based attacks due to the complexity introduced by tabulation-based hashing. However, this security enhancement comes at a cost. It leads to higher computational overhead, as the need to generate and access multiple lookup tables increases processing time. Additionally, there is increased memory consumption because storing large precomputed tables requires additional space. These trade-offs must be considered when choosing between TMH and other privacy-preserving techniques [SAH24; VCRS20]. Despite these trade-offs, TMH remains an attractive alternative for PPRL due to its robustness against adversarial attacks.

Similar to BFs, TMH encodes each record as a bit vector of length l. Given two TMH-encoded bit vectors, their similarity can be estimated using a modified Jaccard coefficient, adapted to account for artificial bit collisions caused by truncation to the least significant bit. The Jaccard coefficient can also be converted into the Dice coefficient for improved comparability with BF-based methods [SAH24; VCRS20].

Overall, TMH provides a more secure encoding alternative to BFs in PPRL, though at the expense of increased computational and memory requirements [SAH24; VCRS20].

2.2.3. TSH

TSH is the most recent encoding scheme proposed for PPRL , introduced in 2020 [RCS20]. TSH was designed to address both the privacy vulnerabilities of BFs and the computational complexity of TMH while maintaining accuracy in similarity calculations. Similar to other encoding techniques, TSH requires the input to be split into a set of n-grams S prior to encoding [RCS20].

As a result of TSH encoding, each record from a sensitive database is represented by a set of integers, which can be directly used to compute Jaccard similarity. TSH employs two distinct hashing steps. In the first hashing step, the input set is converted into a bit matrix representation. In the second hashing step, the bit matrix columns are mapped into integers,

enabling efficient comparison. This two-step process allows TSH to represent sensitive data in a way that facilitates effective similarity computation while preserving privacy [SAH24]. These steps provide accurate Jaccard similarity calculations while improving privacy protection compared to traditional BF-based encodings [RCS20].

In the first step of the TSH process, elements of the n-gram set S are hashed into k independent BFs b_i of length l, meaning each hash function results in a corresponding bit vector. This generates a $k \times l$ matrix, where each row corresponds to a BF created using a unique hash function, and each column represents the bitwise state across all BFs for a given position. In the second step, after constructing the bit matrix, TSH computes column-wise hashes to convert the bit vectors into integer representations . Each column vector is treated as an input for a hash function, and all-zero columns are skipped to prevent distortion in similarity calculations since they do not encode any n-grams. To enhance security and avoid hash collisions between columns with identical bit patterns, a salt value and the column index are concatenated before hashing [RCS20].

The final integer representation for each column i for $1 \leq i \leq l$. is computed as [RCS20; SAH24]:

$$H(salt, i, b_{1i}, b_{2i}, ..., b_{ki}) \tag{2.4}$$

The output of the second hashing step is a set of integers, allowing similarity computations using the Dice coefficient rather than directly computing bitwise similarity. Since the encoded data consists of sets, Jaccard similarity can also be computed similarly to MinHash-based encodings [RCS20].

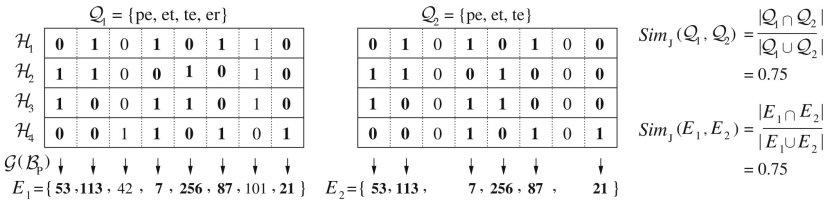

Figure 2.5.: TSH example for for two input values "peter" and "pete" [RCS20].

An illustrative example is provided in Figure 2.5, where the set of 2-grams for the words "*peter*" and "*pete*" is encoded using TSH with $k = 4$ hash functions and a Bloom filter length of $l = 8$. In the first hashing step, each 2-gram is processed using the k hash functions, resulting in a 4×8 bit matrix. Each row of this matrix corresponds to a Bloom filter generated with a distinct hash function, encoding the presence of 2-grams across the l bit positions.

In the second hashing step, the columns of the bit matrix are transformed into integer values. This is achieved by applying an additional hash function that incorporates both a salt value and the column index, effectively compressing the binary representation into a fixed-length

integer vector. The final output is a set of integers representing the encoded word, which can subsequently be used to compute similarity scores between different words [RCS20].

To improve efficiency, TSH can be implemented using a Pseudo-Random Number Generator (PRNG) instead of cryptographic hash functions. The PRNG is seeded with the value to be hashed before generating random numbers, ensuring that the sequence of generated values depends deterministically on the input [RCS20].

By combining efficient bit vector representations with integer-based similarity computations, TSH offers a balance between privacy, security, and computational efficiency, making it a promising alternative to existing PPRL encoding schemes [RCS20; VCRS20].

2.3. GMA

GMAs were introduced by Vidanage et al. [VCRS20] and represent the most significant threat to PPRL due to their universal applicability. Unlike traditional cryptanalytic attacks, GMAs exploit the fundamental properties of non-interactive PPRL to compromise the security of all schemes relying on similarity-preserving encoding [SAH24].

Non-interactive PPRL refers to linkage schemes where data owners independently encode their data and share it with a linkage unit. The linkage unit then performs record matching solely based on the encoded data, without requiring further interaction with the data owners during the linkage process. This approach minimizes communication overhead and computational complexity, as no iterative exchanges between parties are necessary. In contrast, interactive PPRL methods involve multiple rounds of communication between data owners and the linkage unit to refine matching results or improve accuracy [KKM+14].

In PPRL, encoded records are linked based on similarity computations. Since these similarities serve as identifiers, an attacker with access to both encoded and plaintext data can leverage them to re-identify individuals. The latest version of the GMA developed by Schaefer et al. [SAH24] overcomes the limitations of the original attack by Vidanage et al. [VCRS20] and enhances success rate and robustness, even under limited knowledge scenarios [SAH24].

In the context of a GMA on a PPRL system, the attacker is modeled as the linkage unit and is assumed to have minimal prior knowledge. The attacker does not know any encoding secrets, seeds, or salts used in the system to protect the data. The only information available to the attacker is that which is inevitably known to the linkage unit during the linkage process. This assumption ensures that the attacker can only exploit data accessible through normal system operations, adhering to Kerckhoffs's principle [SAH24].

Since the attack does not depend on specific encoding parameters or attribute frequency distributions, it is universally applicable as long as pairwise similarities of encoded data are available [SAH24].

The first step of the attack involves constructing similarity graphs for both the encoded dataset (D_{enc}) and the plaintext dataset (D_{plain}). In these graphs, each node represents an individual record, while edges between nodes are assigned weights based on pairwise similarity computations. To ensure computational efficiency and focus only on meaningful connections, edges with similarity scores below a predefined threshold are omitted, reducing noise and improving the accuracy of the attack [SAH24].

Since certain encoding properties, such as BF length (l), are inevitably known to the linkage unit, D_{plain} can be transformed analogously to D_{enc} for effective comparison. Importantly, this step does not require knowledge of shared secrets, as the primary objective is to replicate

the effect of encoding on similarity [SAH24].

To quantify the structural similarity between nodes in G_{plain} and G_{enc}, node embeddings are computed to transform the graph structure into a numerical representation. This process begins with graph embedding using the Node2Vec algorithm, which applies a Word2Vec-like approach to learn vector representations of nodes. During this process, nodes undergo multiple random walks, where each walk simulates a sequence of transitions between connected nodes. These sequences are then treated as sentences, allowing the model to learn embeddings that capture the local and global structure of the graph. The behavior of these random walks is controlled by two hyperparameters: p, which determines the likelihood of returning to a previously visited node, and q, which influences the tendency to explore new regions of the graph. The result is an embedding matrix where each row represents a node as a vector in Euclidean space [SAH24].

Once embeddings are generated, they must be aligned to allow meaningful comparison between the two graphs. Due to the randomness inherent in embedding generation, direct comparison is not possible. Instead, an iterative approach is used to solve two subproblems: first, an optimal linear transformation is determined using Procrustes Analysis to align the embeddings, and second, node correspondences are established via the Sinkhorn Algorithm, which minimizes the Wasserstein distance between the distributions of embeddings in both graphs. To achieve an effective alignment, an unsupervised stochastic optimization scheme alternates between these two steps over n epochs, gradually refining the transformation and correspondences until convergence [SAH24].

Once the embeddings from the plaintext and encoded datasets are aligned, the reidentification process can begin. Each embedding in the transformed plaintext space is compared to its counterparts in the encoded space, with similarity measured using cosine similarity. This metric quantifies how closely two embeddings align in the high-dimensional space, enabling the attacker to identify records in the encoded dataset that most closely resemble those in the plaintext dataset [SAH24].

The final step involves constructing a bipartite graph, where nodes from the plaintext and encoded datasets are linked based on their similarity scores. To determine the optimal mapping, the Jonker-Volgenant algorithm is applied, ensuring that each node in the smaller dataset is uniquely matched to a corresponding node in the larger dataset. This algorithm maximizes the total similarity across all matched pairs, effectively revealing the identities of individuals within the encoded dataset [SAH24].

An example for this is given in Figure 2.6 and Figure 2.7 where the high-level overview of the GMA attack process is shown for the example of a PPRL approach using BF. First, the two data owners agree on an encoding scheme and encode their respective datasets, for this example it is BF encoding. Then they send their respective encoded dataset to the linkage unit. The attack begins at this point by with the linkage unit leveraging information it inevitably knows. The linakge unit is able to construct similarity graphs for both the encoded datasets and a plaintext datasets. After embedding the similarity gaphs, the embeddings are aligned and the attacker can identify re-identifications by comparing embeddings and constructing a bipartite graph to match entries from one of the encoded datasets with the plaintext dataset [SAH24].

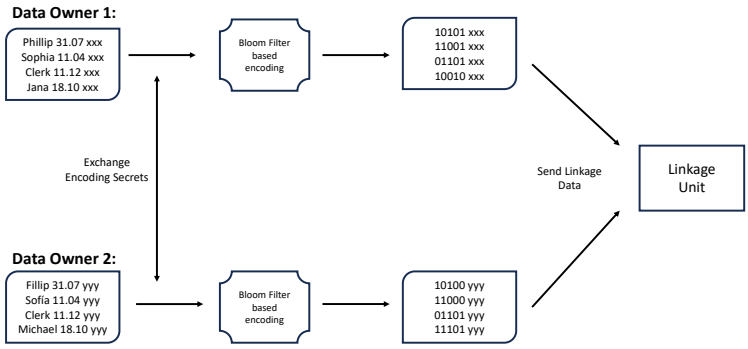

Figure 2.6.: High-level overview of the GMA attack process. Two data owners encode their datasets and send them to the linkage unit.

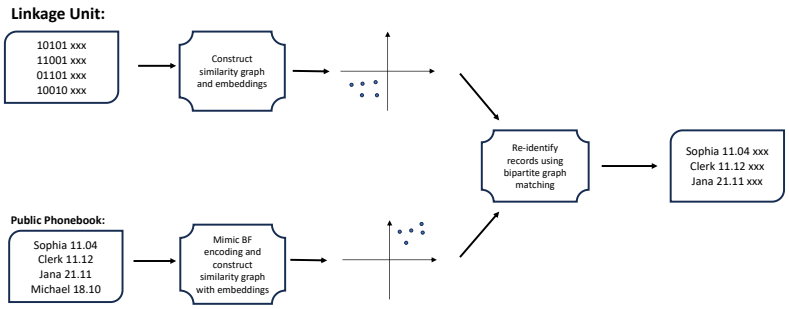

Figure 2.7.: High-level overview of the GMA attack process. The linkage unit mimics the BF encoding for the public dataset and creates for both datasets similartie graphs, embeddings and aligns them to perform bipartite matching.

The novel GMA approach by Schaefer et al. [SAH24] achieves near-perfect re-identification rates when dataset overlap is 100%. Even for low-overlap scenarios (e.g., 5%), success rates reach 99.9% for TSH [SAH24]. The only encoding scheme resistant to GMAs is BFs with diffusion layers, which disrupts similarity preservation for sufficiently high diffusion values [SAH24].

2.4. ANN

ANNs are a class of machine learning models inspired by the structure and function of biological neural systems. They consist of interconnected layers of artificial neurons that process input data and extract meaningful patterns through iterative learning. ANNs have been widely

applied in various fields, including image recognition, natural language processing, and classification tasks. Neural networks are particularly effective for complex tasks because they automatically identify and refine patterns in data through multiple layers of processing, removing the need for manual feature engineering, which can be difficult and time-consuming [DKK+12].

The structure of an ANN consists of multiple layers as can be seen in Figure 2.8, each serving a distinct role in processing and transforming input data. The input layer is the first stage of the network, responsible for receiving raw data and forwarding it to subsequent layers. The number of neurons in this layer corresponds directly to the number of input features, ensuring that all relevant information is passed through the network [DKK+12].

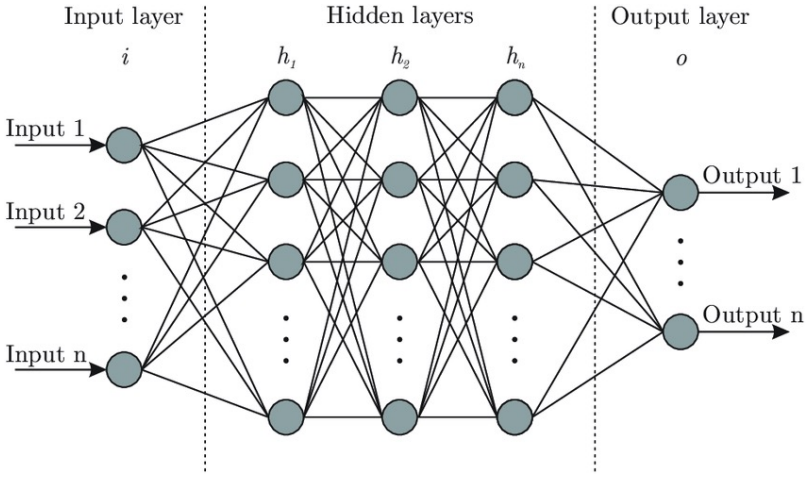

Figure 2.8.: ANN consisting of multiple input neurons (input layer), hidden layers and output neurons (output layer) [BGF17].

Following the input layer are the hidden layers, which perform feature extraction and transformation. Each neuron in a layer applies a weighted sum operation to its inputs, followed by an activation function that introduces non-linearity, enabling the network to learn complex patterns in the data. Common activation functions include the Rectified Linear Unit (ReLU), Sigmoid, Leaky ReLU, and Exponential Linear Unit (ELU), each offering advantages depending on the specific task [RN16; SSA17].

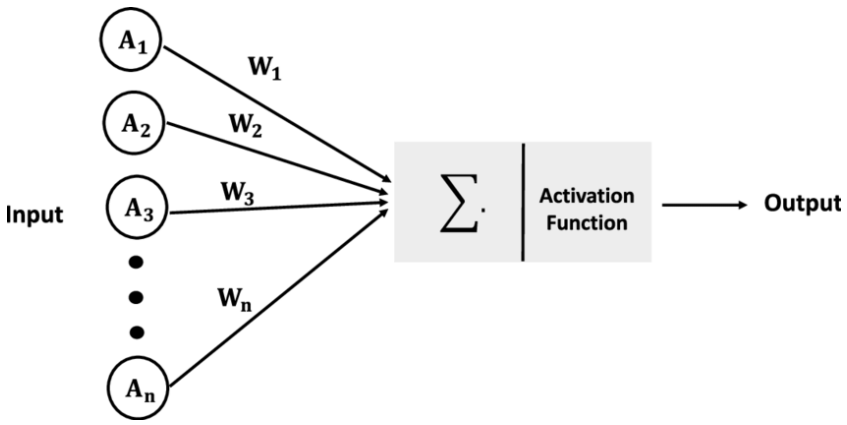

Figure 2.9.: Sketch of an single artifical neuron in an ANN [GSB+21].

An example of this process is illustrated in Figure 2.9, which depicts a single neuron in an ANN. The neuron takes multiple inputs, multiplies them by corresponding weights, sums the weighted inputs along with a bias term, and applies an activation function to compute the final output. Mathematically, this operation is represented as

$$a = \sigma(z) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.5}$$

where a is the neuron's output, σ is the activation function, z is the weighted sum, w_i are the weights, x_i are the inputs, and b is the bias term [GSB+21].

The depth and size of the hidden layers determine the network's capacity to model intricate relationships, making them a crucial component of deep learning architectures [DKK+12].

Finally, the output layer generates the final predictions based on the processed information. The number of neurons in this layer depends on the nature of the task—whether it is a classification problem, where each neuron represents a class, or a regression task, where a single neuron outputs a continuous value [DKK+12].

Training an ANN involves iteratively adjusting its parameters, using a labeled dataset to minimize prediction errors. The process begins with forward propagation, where input data flows through the network, passing through multiple layers until it reaches the output layer, generating a prediction. This prediction is then compared to the actual target value, and the discrepancy between the two is quantified using a loss function, which measures the model's performance [RN16].

To improve accuracy, the network undergoes backward propagation (backpropagation), where the gradient of the loss with respect to each weight is computed using the chain rule of differentiation. These gradients indicate how each parameter should be adjusted to reduce the overall error [RN16].

An optimizer, such as Stochastic Gradient Descent (SGD) or Adam, updates the weights accordingly by taking small steps in the direction that minimizes the loss. The training process is repeated over multiple epochs, where the entire dataset is processed multiple times. To enhance efficiency, the data is often divided into batches, allowing the model to update its weights incrementally rather than processing the entire dataset at once. Over time, this

iterative optimization process enables the network to learn meaningful patterns and improve its predictive performance. ANNs can be applied to different classification tasks, depending on whether a data instance belongs to a single category or multiple categories simultaneously [RN16].

In traditional single-label or binary classification problems, each instance is assigned to one and only one category from a predefined set of classes. To achieve this, the network's output layer typically uses a Softmax activation function, which converts the raw output scores into a probability distribution over all possible classes. The model can be trained using the Cross-Entropy Loss function, which penalizes incorrect classifications by measuring the difference between the predicted probability distribution and the actual class label [HCR+16; RN16].

In contrast, multi-label classification allows an instance to belong to multiple categories at the same time. Instead of a single categorical output, the network produces independent predictions for each possible label. The output layer can then as an example apply sigmoid activations for each label, transforming the raw scores into independent probabilities indicating the presence or absence of each class. Since each label is treated as a separate binary classification problem, Binary Cross-Entropy (BCE) Loss is commonly used to optimize the model, ensuring accurate predictions across multiple labels [HCR+16; RN16].

Different ANN architectures have been developed to address various problem domains, each optimized for specific types of data and tasks. Feedforward ANNs (FNNs) represent the simplest architecture, where data flows in one direction from the input layer to the output layer without forming cycles. These networks are widely used for basic classification and regression tasks but may struggle with complex patterns that require spatial or sequential dependencies [GB10; RN16].

For tasks involving image processing, Convolutional ANNs (CNNs) are commonly used. CNNs employ convolutional layers that apply filters to input images, allowing the network to capture complex patterns such as edges, textures, and shapes. This makes them highly effective for applications like object recognition and medical imaging [ON15].

When dealing with sequential data, Recurrent ANNs (RNNs) and their advanced variant, Long Short-Term Memory (LSTM) Networks, are particularly useful. These architectures introduce recurrent connections, enabling them to maintain memory of previous inputs and recognize patterns over time. This makes them well-suited for natural language processing, speech recognition, and time series forecasting [MJ+01].

2.4.1. PyTorch for Training ANN

Training ANNs in PyTorch involves defining and optimizing a model through iterative learning processes while leveraging GPU acceleration for efficient computation. PyTorch provides a flexible framework for designing and training neural networks, making it a popular choice for deep learning research and applications [Fou].

The training process begins with model definition, where the architecture, including layers, activation functions, and parameters, is specified using the ‘torch.nn.Module‘ class. This allows users to define complex networks with full control over forward propagation. Next, a loss function is chosen based on the task at hand [Fou].

Once the model and loss function are defined, an optimizer is selected to update the model's weights during training. Training proceeds over multiple epochs, where the dataset is processed in several passes to refine the model's performance. To further optimize learning, the dataset is divided into mini-batches, allowing for efficient gradient updates without requiring full dataset

processing at once [Fou].

One of PyTorch's key advantages is its support for GPU acceleration via 'torch.cuda', which significantly reduces training time for large datasets. By moving tensors and models to a GPU, computations are performed in parallel, leading to substantial performance gains compared to CPU-based training. Additionally, PyTorch's autograd engine enables automatic differentiation, simplifying the backpropagation process and making model optimization more efficient [Fou].

Despite their success, ANNs present several challenges that must be carefully managed to ensure robust and efficient learning. One of the most common issues is overfitting, where a model becomes too specialized in learning patterns from the training data, capturing possible noise rather than generalizable features. This leads to poor performance on unseen data. Techniques such as dropout regularization, L2 weight decay, and early stopping are commonly used to mitigate overfitting and improve generalization [Fou; GB10].

Dropout regularization is a technique used to prevent overfitting by randomly setting a fraction of neurons to zero during training. This forces the network to learn more robust features by preventing it to rely too heavily on a single neuron to perform well. L2 weight decay is another regularization method that penalizes large weights by adding a regularization term to the loss function. It tries to prevent the network from applying too much importance to a single feature, encouraging it to learn more generalizable patterns. Early stopping is a simple yet effective technique that stops training when the model's performance on a validation set starts to degrade below a certain threshhold, preventing overfitting [Fou; GB10].

Another fundamental challenge is the vanishing and exploding gradient problem, which occurs in deep networks during backpropagation. When gradients become too small (vanishing), weight updates diminish, leading to slow or stalled learning. This can happen because gradients are the product of multiple derivatives, whichh can cause them to shrink exponentially as they propagate through the network. This can happen especially using activiation functions that saturate for extreme values. Conversely, when gradients grow too large (exploding), unstable updates cause erratic training behavior. Solutions such as batch normalization, gradient clipping, and advanced activation functions like Leaky ReLU help address these issues [Fou; GB10].

Batch normalization normalizes the input for each layer ensuring that values are staying in a stable range. Gradient clipping is a technique used to prevent exploding gradients by setting a threshold for the gradient to prevent them from destabilizing the training process. Leaky ReLU is an activation function that prevents the vanishing gradient problem by allowing a small gradient for negative values, ensuring that the network can continue learning even for extreme inputs [Fou; GB10].

The computational complexity of deep learning models is another major concern, as large-scale ANNs require extensive memory and processing power. Training deep networks on large datasets can be prohibitively slow on CPUs, necessitating the use of GPUs or specialized hardware like TPUs (Tensor Processing Units) to accelerate training. Efficient data-loading techniques and mixed-precision training can further optimize computational efficiency. PyTorch leverages this by utilizing its DataLoader class to efficiently load and preprocess data by using multiple workers. Mixed-precision training is a technique that partly uses lower-precision floating-point numbers to reduce memory usage and speed up computations, while still maintaining model accuracy [Fou].

Lastly, hyperparameter tuning plays a critical role in model performance. Selecting the right learning rate, batch size, number of layers, and optimization algorithm requires experimenta-

tion and fine-tuning. Automated methods such as grid search, random search, and Bayesian optimization can assist in finding optimal configurations, but these processes are computationally expensive. Addressing these challenges effectively is crucial for developing high-performing ANNs that generalize well across different datasets and tasks [Fou].

ANNs have revolutionized the field of machine learning by providing better models capable of solving complex tasks. Their flexibility allows them to be applied to various domains, from image recognition to attacking privacy-preserving record linkage systems. With frameworks like PyTorch, researchers and practitioners can leverage GPU acceleration to train large-scale models efficiently [Fou].

3. Methodology

The DEA is a novel attack method that extends the capabilities of GMAs by moving beyond the intersection of datasets to re-identify individuals who were previously unmapped. This chapter outlines the methodology behind the DEA, including modifications to the GMA, the design and implementation of the DEA itself, and the use of ANNs to enable probabilistic reconstruction of PII from encoded data.

The DEA builds upon the GMA by using its re-identification results as a foundation for further inference. While the GMA re-identifies only those records that exist in both the attacker's dataset and the encoded target dataset, it often leaves a substantial portion of records unmapped. The goal of the DEA is to extend this re-identification process by applying a machine learning–based approach to infer the missing PII of the remaining records using a trained ANN.

To achieve this, the DEA follows a structured pipeline comprising six key steps, as illustrated in Figure 3.1. The first step involves executing the GMA and extracting its results in a predefined format to serve as training data. This dataset includes the re-identified individuals, their corresponding encoded representations, and the plaintext information that was successfully linked. In addition, the GMA results contain the non-re-identified individuals, who are represented solely by their encoded PII and associated plaintext values.

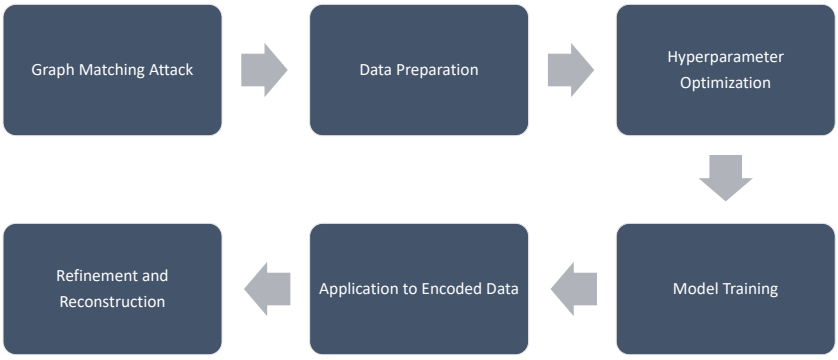

Figure 3.1.: Overview of the DEA attack pipeline.

Once the data is extracted, it undergoes a transformation process to prepare it for ANN training. This data preparation step involves constructing specialized datasets that convert the encoded representations and their corresponding labels—plaintext n-grams—into tensor-based formats suitable for processing by deep learning models. The resulting datasets are subsequently split into training, validation, and test subsets, and corresponding data loaders are created to facilitate efficient mini-batch processing during model training.

With the data pipeline in place, a hyperparameter optimization is conducted to identify the best-performing model configuration. This process systematically evaluates various combinations of hyperparameters, including the number of hidden layers, layer widths, activation functions, optimizers, and learning rate schedulers. The optimal hyperparameters are then used to define the ANN architecture, which is trained to learn the mapping between encoded representations and their corresponding plaintext n-grams.

The architecture of the ANN is tailored to the specific characteristics of the encoding schemes used in the PPRL scheme. The input layer size is determined by the dimensionality of the encoded representation, while the output layer size corresponds to the size of the predefined n-gram dictionary.

Once the best hyperparameter configuration is identified, the ANN is trained using the re-identified individuals as labeled data. Training proceeds over multiple epochs, during which the model iteratively processes the training dataset, computes the loss, and updates its parameters via backpropagation. Performance is continuously monitored on the validation set to track generalization and prevent overfitting.

Once the ANN is trained, it can be applied to the set of non-re-identified individuals, i.e., records that remained unmapped after the GMA. This dataset serves as the test set during the experimentation phase to evaluate the performance of the attack.

The model outputs a probability distribution over all possible n-grams for each entry, indicating the likelihood of each n-gram being present in the corresponding plaintext data. To refine these predictions, a thresholding mechanism is applied to filter out low-confidence outputs and retain only the most probable n-grams. These predicted n-grams are then aggregated and reconstructed into potential PII, constituting the final step of the DEA process.

This methodological approach represents a significant advancement in attacking PPRL systems. By leveraging deep learning techniques, the DEA enables an attacker to infer sensitive personal information beyond the scope of traditional GMA approaches. The following sections provide a detailed discussion of each component, including the design choices, implementation details, and challenges encountered during development.

3.1. Modifications to the GMA

The GMA serves as the foundation for the DEA by providing an initial set of re-identified individuals, including their corresponding encodings, as well as the remaining non-re-identified individuals.

The effectiveness of the DEA is directly influenced by the performance of the GMA. A higher re-identification rate in the GMA yields a larger training dataset for the DEA, thereby enhancing its ability to infer n-grams and re-identify individuals in the non-re-identified dataset. Conversely, a low re-identification rate limits the availability of labeled training data, reducing the overall effectiveness of the DEA.

To integrate the GMA as a preprocessing step for the DEA, modifications were made to the original implementation by Schaefer et al. [SAH24]. While the core algorithm remains unchanged, adjustments were introduced to ensure that the GMA outputs its results in a structured format suitable for training the ANN used in the DEA.

Originally, the GMA only provided a simple mapping between the IDs of re-identified individuals. However, to enable the DEA to learn meaningful patterns, access to both the plaintext PII and their corresponding encodings is required. Therefore, the GMA was extended to output

two datasets in the following format:

- For re-identified individuals: `<PII> <encoding> <uid>`
- For non-re-identified individuals: `<encoding> <uid>`

It is important to note that the `uid` is included solely for research and evaluation purposes. It enables researchers to manually track individuals across different processing stages and to assess the performance of the attack. However, in a real-world attack scenario, these `uid`s are neither available nor required. They are entirely excluded from all DEA training and inference steps, ensuring that the attack methodology remains realistic and practically applicable.

In addition to formatting adjustments, certain components of the GMA were removed to streamline the process and reduce unnecessary complexity. Specifically, encoding schemes other than TSH, TMH, and BF were excluded, as the DEA focuses exclusively on these techniques. Other components deemed non-essential — such as graph visualizations and benchmark tests related solely to the GMA — were also removed. This decision was made because the GMA is not the primary focus of this study; its validity and performance have already been established by prior research. These optimizations resulted in a leaner and more efficient attack pipeline, reducing computational overhead while preserving essential functionality.

With these modifications in place, the starting point for the DEA is clearly defined. The attack begins with two structured datasets:

1. Re-identified individuals, containing both plaintext PII and corresponding encodings, formatted as described above.
2. Non-re-identified individuals, for whom only the encodings are available, serving as the primary targets for inference using the DEA.

By leveraging this structured output, the DEA can train a machine learning model to probabilistically reconstruct missing n-grams from the encoded records of non-re-identified individuals. The following sections detail the implementation of this approach, including dataset preparation, model architecture, and evaluation strategies.

3.2. Design and Implementation of the DEA

The DEA aims to reconstruct plaintext PII from encoded records using machine learning techniques. This section outlines the problem definition, data representation, ANN architecture, and training methodology. A central challenge in implementing the DEA lies in the diversity of encoding schemes used to protect sensitive data. As each encoding method transforms plaintext into distinct numerical representations, the DEA must adapt both the dataset structure and the ANN architecture accordingly.

To address this, the DEA adopts a modular design: while the overall attack methodology remains consistent, specific implementations are tailored to each encoding scheme. Although the input representation and network architecture vary depending on the encoding, the output format is kept uniform across all models. The attack is framed as a multi-label classification task, where the ANN predicts the likelihood of individual n-grams appearing in the original PII. For each encoding scheme, a dedicated dataset structure transforms encoded records into a format suitable for ANN training. Moreover, a custom ANN architecture is employed for

each encoding scheme to ensure the model effectively learns the mapping from encoded data to plaintext n-grams.

3.2.1. Problem Definition

The primary challenge that the DEA seeks to address is the limited scope of re-identifications achieved by the GMA. While the GMA effectively links records by exploiting structural relationships within encoded datasets, its success is inherently restricted to individuals who are present in both the plaintext and the encoded datasets and can be matched based on graph similarity. However, in real-world scenarios, there may exist additional re-identification potential beyond these direct matches.

One possible way to extend re-identifications is to rerun the GMA iteratively, incorporating additional publicly available data to gradually refine the matching process. However, this approach is inherently dependent on the availability and quality of external data sources, which may not always be feasible. Instead, the DEA introduces a novel strategy that aims to reconstruct deterministic relationships between encoded representations and their corresponding plaintext information. This is based on the observation that all encoding schemes used in PPRL rely on hash functions or other deterministic mappings.

Hash functions, for example, produce fixed-length outputs from inputs of arbitrary length and, crucially, are deterministic — meaning the same input will always yield the same output. The DEA leverages this property by training ANNs to learn statistical relationships between encoded values and the original n-grams of PII. The objective is to recover the most probable plaintext representation given an encoded input, effectively framing the attack as a probabilistic, frequency-based inference problem. However, several challenges complicate this task.

The first major challenge is the lack of knowledge about the specific number and type of hash functions used during encoding. As a result, the model must learn patterns in the data without any explicit understanding of the underlying hashing mechanisms. Fortunately, this limitation is partially mitigated by the fact that the DEA does not rely on a one-to-one mapping between hash outputs and plaintext n-grams, but instead depends on statistical inference across large numbers of training samples.

A more fundamental challenge arises from the collision property of hash functions. Because hash functions map an infinite input space to a finite output space, different inputs may produce identical hash values, making it inherently difficult to perfectly recover the original plaintext. These collisions introduce uncertainty into the re-identification process, preventing the DEA from achieving perfect reconstruction accuracy. Consequently, the predictions made by the DEA are probabilistic rather than deterministic — it can estimate the likelihood of a specific n-gram being present in the original PII, but cannot guarantee absolute correctness.

The primary reason the GMA alone is unable to achieve such probabilistic reconstruction is that it relies solely on structural similarities within the dataset, without attempting to infer direct relationships between encoded values and their plaintext equivalents. In contrast, the DEA enhances the capabilities of the GMA by enabling the reconstruction of individual plaintext components directly from encoded representations, thereby increasing the overall re-identification potential. This novel approach significantly improves the effectiveness of the attack, allowing for the re-identification of individuals who were previously unmatchable using traditional graph-based techniques.

3.2.2. Data Representation

For an ANN to operate effectively, the input data must be preprocessed into a format compatible with deep learning models. This preprocessing step applies to both the input — encoded representations of PII — and the output, which consists of labels representing the predicted n-grams. Since ANNs in PyTorch operate on tensor-based representations, the transformation of encoded records into tensors is a fundamental requirement. This ensures that both re-identified and not-reidentified individuals are structured in a way that enables efficient training and inference.

To facilitate this transformation, custom PyTorch datasets are implemented. These datasets convert encoded representations into input tensors and encode the corresponding n-gram labels in a multi-label classification format, where each n-gram is represented as a binary indicator within a fixed-size label vector. This approach enables the model to predict the presence of multiple n-grams per encoded record.

The data representation pipeline is modular and accommodates various encoding schemes, each of which necessitates a tailored preprocessing technique. Depending on the encoding method — such as BF, TSH, or TMH — different strategies are employed to convert the encoded input into tensors while preserving as much discriminative information as possible. This ensures that the input is well-suited to the architecture of the corresponding ANN and that the model can effectively learn the mapping between encoded data and plaintext n-grams.

3.2.2.1. BF Encoding

BF's are fixed-length binary strings, with their length determined by Alice's chosen encoding parameters. The transformation of a BF into a PyTorch tensor is straightforward: each bit in the binary string is directly mapped to a corresponding position in the tensor. This conversion preserves the positions of set bits (i.e., ones), thereby maintaining the structural integrity of the original encoding. The resulting tensor has the same dimensionality as the BF, with ones indicating the activated hash positions and zeros elsewhere. This binary representation serves as the input to the ANN, allowing the model to learn patterns based on the bitwise structure of the encoded PII.

3.2.2.2. TMH Encoding

TMH, like BFs, produces fixed-length binary bitstrings, with the specific length determined by the encoding parameters selected by Alice. The transformation into a PyTorch tensor mirrors that of the BF: each bit in the TMH string is mapped directly to a corresponding tensor position, preserving the locations of set bits. This direct conversion results in a binary tensor representation that retains the structure of the original TMH encoding. By preserving the positional information of the activated bits, the ANN can effectively learn from the encoded patterns embedded in the TMH representations.

3.2.2.3. TSH Encoding

The preprocessing of TSH encodings is more complex due to its variable-length representation. Unlike BF and TMH, which produce fixed-length binary bitstrings, TSH generates a set of integers of arbitrary size. This variability arises because columns containing only zero values are dropped during the TSH encoding process.

Since ANNs require fixed-length input vectors, a suitable transformation must be applied to standardize TSH encodings. Aggregation techniques, such as computing averages, would result in significant information loss, especially this already limited knowledge setting. Therefore, an alternative approach is employed to convert TSH encodings into a tensor-compatible format.

To achieve this, all unique integer values from both the re-identified and non-reidentified datasets are collected and stored in a set. This set is then sorted in ascending order and transformed into a dictionary that maps each integer to a unique index. Using this mapping, each TSH encoding is converted into a binary vector using a one-hot encoding scheme. For each integer present in the TSH encoding, the corresponding index in the binary vector is set to one, while all other positions remain zero.

Regardless of the encoding scheme used as input, the output of the ANN remains consistent across all implementations. The model is trained to map the encoded input to a probability distribution over possible n-grams. Thus, the output layer of the ANN performs multi-label classification, predicting the likelihood of each n-gram being present in the original plaintext PII.

3.2.3. Re-Identified Individuals as Labeled Training Data

To enable supervised learning, re-identified individuals are used as labeled training and validation data. Since their PII is known along with their corresponding encoded representation, it is possible to construct datasets where the input consists of transformed encodings (BF, TMH, or TSH, respectively) and the output labels consist of the correct n-grams derived from the original PII.

To facilitate this process, a predefined dictionary of all possible n-grams is created. This dictionary includes:

- Alphabetical n-grams (e.g., for 2-grams: `aa` to `zz`),
- Numerical n-grams (e.g., for 2-grams: `00` to `99`),
- Alphanumeric mixed n-grams (e.g., for 2-grams: `a0` to `z9`).

Since the datasets used in this research primarily contain first names, last names, and birthdates, these character sets are sufficient to cover the vast majority of n-gram occurrences. Each possible n-gram is mapped to a specific index in the output tensor based on the dictionary, ensuring a consistent label format across all training samples. For example, if index `1` corresponds to the 2-gram "ab", and the ANN predicts a 60% probability at index `1`, this is interpreted as a 60% likelihood that "ab" was present in the original plaintext.

By structuring the data in this way, the ANN is trained to learn a mapping from encoded inputs to their corresponding n-gram distributions, enabling the DEA to probabilistically reconstruct plaintext PII from encoded data.

3.2.4. ANN Architecture for DEA

Attempting to reconstruct plaintext information from encoded representations based on hash functions presents a significant challenge due to the nature of cryptographic hashing. Since hash functions are designed as one-way functions, reversing the transformation to recover the original input is theoretically infeasible. However, while exact reconstruction is not possible,

a probabilistic approach can still be employed to infer likely plaintext components based on statistical patterns within the encoded data.

ANNs provide a powerful framework for learning complex mappings between input encodings and output predictions, making them well-suited for this task. The function of the ANN in the context of the DEA is to predict n-grams by learning from re-identified individuals — those whose plaintext information is known alongside their corresponding encodings. Through this supervised learning process, the model captures frequency patterns that emerge due to the deterministic nature of the encoding process. In essence, the ANN learns which n-grams are statistically associated with specific positions or patterns in the encoded representations and leverages these associations to estimate their likelihood in unseen encoded inputs.

Although hash functions introduce collisions, where different inputs may produce the same hash output, the ANN can still extract meaningful probabilistic insights by generalizing over these mappings across many samples. This enables the DEA to output a ranked list of likely n-grams per record, thereby forming the basis for the reconstruction of PII from encoded data in a probabilistic, frequency-informed manner.

3.2.4.1. General ANN Architecture

To support the task of reconstructing n-grams from encoded data, a flexible ANN architecture is employed across all models. This architecture is designed to adapt to the different input formats arising from various privacy-preserving encoding schemes. Although the input representation differs depending on the encoding used, the output layer remains the same across all trained models. The output is a probability distribution over a fixed dictionary of n-grams, where each output neuron corresponds to the likelihood of a specific n-gram being part of the original plaintext string.

The architecture follows a feedforward design and consists of three main components: an input layer, a configurable sequence of hidden layers, and a final output layer. The size of the input layer is determined by the dimensionality of the encoded record. For instance, in the case of the BF and TMH model, the input layer corresponds to the length of the bitstring, which in turn is defined by Alice's chosen parameters. The TSH models define their input layers based on the number of unique integers used across both datasets.

This modular architecture enables experimentation across different encoding schemes while maintaining a unified framework for training and evaluation.

The hidden layers are structured dynamically, depending on a set of tunable hyperparameters. These include the number of hidden layers, the number of neurons in each layer (hidden layer size), the activation function (e.g., ReLU, Tanh), and the dropout rate used for regularization. Each hidden layer is followed by a non-linear activation function, enabling the model to capture complex and non-linear relationships in the data. To mitigate overfitting, dropout is applied after each activation layer, randomly deactivating a fraction of neurons during training. This regularization technique improves the model's ability to generalize to unseen inputs and prevents the memorization of training data.

The output layer remains consistent across all encoding schemes and has a dimensionality equal to the size of the predefined n-gram dictionary. Each output neuron represents the model's predicted probability that a specific n-gram appears in the original plaintext record. Given that multiple n-grams may be present in a single encoded record, the task is framed as a multi-label classification problem. Therefore, the output layer uses a sigmoid activation function, allowing the network to assign independent probability estimates to each n-gram.

This modular architecture is implemented using PyTorch's `nn.Sequential` API, which enables a clean, maintainable, and extensible model definition. Furthermore, this design supports efficient hyperparameter optimization, as critical components—such as the number of layers, hidden layer size, activation function, and dropout rate—can be systematically varied across experimental runs. By exploring this hyperparameter space, the model can be tailored to maximize reconstruction performance for each specific encoding scheme.

3.2.4.2. Hyperparameter Optimization

Hyperparameter tuning plays a crucial role in achieving optimal model performance. Unlike model parameters that are learned during training (e.g., weights and biases), hyperparameters are defined prior to training and control the structure of the model as well as aspects of the learning algorithm. These include architectural choices such as the number of layers as well as training configurations like the learning rate, optimizer, and regularization techniques. Careful selection of these values is especially important in complex tasks such as reconstructing plaintext n-grams from encoded representations, where both underfitting and overfitting can lead to substantial performance degradation.

To explore the extensive hyperparameter space efficiently, this work employs Ray Tune, a scalable library for distributed hyperparameter tuning. Specifically, the Optuna search algorithm is used within Ray Tune to guide the optimization process. Optuna leverages a Tree-structured Parzen Estimator, a Bayesian optimization method that prioritizes promising regions of the search space based on previous trial results. This approach improves search efficiency and reduces the number of iterations required to discover high-performing configurations.

The hyperparameter search space in this study is designed to be both comprehensive and computationally feasible. Key hyperparameters that define the neural network architecture include:

- **Number of hidden layers** (`num_layers`): varied between 1 and 7, allowing the exploration of both shallow and deep networks.
- **Hidden layer size** (`hidden_layer_size`): selected from {64, 128, 256, 512, 1024, 2048}, enabling experiments with compact to large-capacity models.
- **Dropout rate** (`dropout_rate`): sampled uniformly between 0.1 and 0.4 to promote generalization and mitigate overfitting.
- **Activation function** (`activation_fn`): treated as a categorical variable with options including `ReLU`, `Leaky ReLU`, `GELU`, `ELU`, `SELU`, and `Tanh`.

These architectural parameters create a flexible and expressive search space for discovering well-performing network structures tailored to the task of the DEA.

The optimization strategy is similarly governed by several hyperparameters that influence how the model is trained. The **optimizer** is treated as a categorical hyperparameter, with options including `Adam`, `AdamW`, `RMSprop`, and `SGD`. Each optimizer is paired with a corresponding learning rate sampled from a log-uniform distribution to accommodate the wide sensitivity of models to this parameter. In the specific case of `SGD`, an additional `momentum` parameter is also tuned to control the influence of past gradients in the current weight update.

In conjunction with the optimizer, the choice of a **learning rate scheduler** (`lr_scheduler`) further enhances the model's ability to converge effectively. The search space for learning rate scheduling strategies includes:

- `StepLR`: reduces the learning rate at fixed epoch intervals,
- `ExponentialLR`: applies exponential decay over time,
- `ReduceLROnPlateau`: reacts to stagnation in validation loss,
- `CosineAnnealingLR`: follows a cosine decay schedule,
- `CyclicLR`: oscillates between lower and upper bounds in modes such as `triangular`, `triangular2`, and `exp_range`.

An additional option to disable learning rate scheduling is also included to assess whether constant learning rates perform better for certain models.

Furthermore, the **loss function** (`loss_fn`) is a critical hyperparameter, as it directly influences the optimization objective. Several loss functions suitable for multi-label classification are explored:

- `BCEWithLogitsLoss`: a standard binary cross-entropy loss combined with a sigmoid activation,
- `WeightedBCE`: a weighted variant to address class imbalance,
- `MultiLabelSoftMarginLoss`: supports probabilistic multi-label targets,
- `FocalLoss`: down-weights easy examples to focus learning on hard-to-predict samples, particularly effective in imbalanced scenarios.

This comprehensive optimization configuration enables systematic exploration of training dynamics, ensuring the neural network can effectively learn meaningful mappings for the Dataset Extension Attack across various encoding schemes.

Additional parameters are also incorporated into the hyperparameter search to fine-tune the model's output behavior and ensure consistent evaluation. A tunable **threshold** parameter is introduced, ranging between 0.3 and 0.8, which is used to convert the model's output probabilities into binary predictions for the presence of specific n-grams. This threshold is particularly relevant in multi-label classification settings where choosing an appropriate cutoff directly impacts precision and recall.

The dimensions of the input and output layers, denoted by `input_dim` and `output_dim`, are fixed for each encoding scheme. The `input_dim` reflects the length of the encoded bitstring or vector depending on the method used (e.g., BF, TMH, or TSH), while the `output_dim` corresponds to the size of the n-gram dictionary, representing the total number of possible n-grams to be predicted.

To ensure a fair and consistent comparison across all hyperparameter configurations, the same `data_train` and `data_val` datasets are used in each trial. This controlled setup ensures that variations in performance can be attributed to the model configuration rather than differences in training and validation data.

During tuning, Ray Tune orchestrates multiple parallel trials, each corresponding to a unique combination of hyperparameters sampled from the search space. Optuna's pruning mechanism

is also integrated, allowing unpromising trials to be stopped early based on intermediate results (e.g., validation loss), which improves overall efficiency. Performance is evaluated on the validation set, and the best configuration is selected based on a predefined objective function.

This automated, systematic tuning process ensures that the neural network architecture is well-adapted to the complexity and characteristics of the input encoding. It enables fair comparison across models (BF, TMH, or TSH) and improves both the predictive performance and generalization capability of the reconstruction task.

3.2.5. Training the Model

To effectively train and evaluate the ANN model with the best hyperparameters, the dataset is divided into three distinct subsets: a training set, a validation set, and a test set. The training set consists of 80% of the labeled dataset, while the remaining 20% is designated as the validation set. The test set comprises the not-reidentified individuals, serving as the primary evaluation set for the trained model.

Dataloaders are created for each of these subsets to facilitate efficient mini-batch processing. Different batch sizes are employed depending on the dataset subset to optimize computational performance and convergence behavior. The training and validation dataloaders enable efficient iteration over the respective data splits, ensuring that the ANN is exposed to all available samples during training and validation.

The training process consists of multiple epochs, where each epoch involves iterating through the entire training dataset using the data loader. For each mini-batch, the model performs a forward pass, computes the loss, and applies backpropagation to update the network's parameters using the selected optimizer. If a learning rate scheduler is specified, it is applied according to its strategy to dynamically adjust the learning rate throughout training.

After processing all training batches in an epoch, the model's performance is evaluated on the validation set by computing the validation loss and selected performance metrics. This allows the training loop to monitor potential overfitting and adjust training accordingly, for instance by applying early stopping or learning rate decay. Throughout this process, the model's weights that achieve the best validation score according to the predefined objective (e.g., dice coefficient) are stored and later used for evaluation on the test set.

3.2.5.1. Loss Computation and Performance Metrics

The performance of the ANN models is assessed using multiple evaluation criteria. The primary metrics include training loss, validation loss, precision, recall, F1-score, and the Dice similarity coefficient.

During training, the loss is computed for each mini-batch, and the cumulative loss across the entire training dataset is used to track the optimization progress. Similarly, the validation loss is computed over the validation set at the end of each epoch to monitor the model's generalization capability. Both training and validation loss curves can be analyzed post hoc to gain further insights into convergence behavior and potential overfitting. The validation loss is also used to determine early stopping, halting training if no improvement is observed for a predefined number of epochs and a specified minimum delta threshold.

Precision quantifies the proportion of correctly predicted 2-grams among all predicted 2-grams. It reflects the model's ability to avoid false positives during the reconstruction of plaintext features from different encoding schemes. A high precision score indicates that most

of the predicted n-grams are indeed part of the original token, suggesting a low rate of over-generation. This is especially important in the context of dataset extension attacks, where the specificity of the reconstructed values is critical.

Recall captures the proportion of true 2-grams that have been successfully predicted by the model. It measures the completeness of the reconstruction, indicating how many of the original n-grams were recovered from the encoded representation. A high recall implies that the attack is able to extract a substantial portion of the underlying information, albeit potentially at the cost of including false positives.

The **F1-score** represents the harmonic mean of precision and recall, offering a balanced metric that integrates both correctness and completeness. This is particularly useful when the number of predicted and ground-truth n-grams differs, as it prevents either metric from disproportionately influencing the overall evaluation.

The **Dice similarity coefficient**, which is mathematically equivalent to the F1-score for binary sets, is computed as twice the size of the intersection of the predicted and actual n-gram sets divided by the total number of elements in both sets. In the DEA setting, the Dice coefficient serves as an interpretable and robust measure of set overlap. It is especially well-suited for evaluating partial reconstructions, where perfect recovery may be infeasible, but significant alignment with the ground truth still reflects successful inference.

Furthermore, a re-identification rate is determined ...

4. Results

4.1. Experiments

4.2. Evaluation Metrics

4.3. Analysis

4.4. Discussion

5. Conclusion

5.1. Summary

5.2. Future Work

Bibliography

[AHS23] Frederik Armknecht, Youzhe Heng, and Rainer Schnell. "Strengthening privacy-preserving record linkage using diffusion." In: *Proceedings on Privacy Enhancing Technologies* (2023).

[BGF17] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks." In: *Energy and Buildings* 158 (Nov. 2017). DOI: 10.1016/j.enbuild.2017.11.045.

[Blo70] Burton H Bloom. "Space/time trade-offs in hash coding with allowable errors." In: *Communications of the ACM* 13.7 (1970), pp. 422–426.

[Bro97] Andrei Z Broder. "On the resemblance and containment of documents." In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE. 1997, pp. 21–29.

[DKK+12] AD Dongare, RR Kharde, Amit D Kachare, et al. "Introduction to artificial neural network." In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194.

[FS69] Ivan P Fellegi and Alan B Sunter. "A theory for record linkage." In: *Journal of the American Statistical Association* 64.328 (1969), pp. 1183–1210.

[Fou] The Linux Foundation. *PyTorch — pytorch.org*. https://pytorch.org. [Accessed 12-03-2025].

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[GSB+21] Fardin Ghorbani, Javad Shabanpour, Sina Beyraghi, Hossein Soleimani, Homayoon Oraizi, and M. Soleimani. "A deep learning approach for inverse design of the metasurface for dual-polarized waves." In: *Applied Physics A* 127 (Nov. 2021), p. 869. DOI: 10.1007/s00339-021-05030-6.

[HCR+16] Francisco Herrera, Francisco Charte, Antonio J Rivera, María J Del Jesus, Francisco Herrera, Francisco Charte, Antonio J Rivera, and María J del Jesus. *Multilabel classification*. Springer, 2016.

[HSW07] Thomas N Herzog, Fritz J Scheuren, and William E Winkler. *Data quality and record linkage techniques*. Vol. 1. Springer, 2007.

[IH18] Jim Isaak and Mina J Hanna. "User data privacy: Facebook, Cambridge Analytica, and privacy protection." In: *Computer* 51.8 (2018), pp. 56–59.

[KKM+14] Hye-Chung Kum, Ashok Krishnamurthy, Ashwin Machanavajjhala, Michael K Reiter, and Stanley Ahalt. "Privacy preserving interactive record linkage (PPIRL)." In: *Journal of the American Medical Informatics Association* 21.2 (2014), pp. 212–220.

[KM24] Jennifer King and Caroline Meinhardt. *Rethinking Privacy in the AI Era: Policy Provocations for a Data-Centric World*. White Paper, Stanford University Institute for Human-Centered Artificial Intelligence (HAI). 2024. URL: http://www.darkpatternstipline.org.

[MJ+01] Larry R Medsker, Lakhmi Jain, et al. "Recurrent neural networks." In: *Design and Applications* 5.64-67 (2001), p. 2.

[MK19] Karl Manheim and Lyric Kaplan. "Artificial intelligence: Risks to privacy and democracy." In: *Yale JL & Tech.* 21 (2019), p. 106.

[ON15] Keiron O'shea and Ryan Nash. "An introduction to convolutional neural networks." In: *arXiv preprint arXiv:1511.08458* (2015).

[PSZ+24] Aditi Pathak, Laina Serrer, Daniela Zapata, Raymond King, Lisa B Mirel, Thomas Sukalac, Arunkumar Srinivasan, Patrick Baier, Meera Bhalla, Corinne David-Ferdon, et al. "Privacy preserving record linkage for public health action: opportunities and challenges." In: *Journal of the American Medical Informatics Association* 31.11 (2024), pp. 2605–2612.

[RCS20] Thilina Ranbaduge, Peter Christen, and Rainer Schnell. "Secure and accurate two-step hash encoding for privacy-preserving record linkage." In: *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part II 24*. Springer. 2020, pp. 139–151.

[RN16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. pearson, 2016.

[SAH24] Jochen Schäfer, Frederik Armknecht, and Youzhe Heng. "R+R: Revisiting Graph Matching Attacks on Privacy-Preserving Record Linkage." In: *Proceedings of [Conference Name, if available]*. Available at: https://github.com/SchaeferJ/graphMatching. University of Mannheim. 2024.

[SBR09] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." In: *BMC medical informatics and decision making* 9 (2009), pp. 1–11.

[SSA17] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks." In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[Smi16] Tanshanika T Smith. *Examining data privacy breaches in healthcare*. Walden University, 2016.

[Smi17] Duncan Smith. "Secure pseudonymisation for privacy-preserving probabilistic record linkage." In: *Journal of Information Security and Applications* 34 (2017), pp. 271–279.

[VCRS20] Anushka Vidanage, Peter Christen, Thilina Ranbaduge, and Rainer Schnell. "A graph matching attack on privacy-preserving record linkage." In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 1485–1494.

[VSCR17] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. "Privacy-preserving record linkage for big data: Current approaches and research challenges." In: *Handbook of big data technologies* (2017), pp. 851–895.

A. Auxiliary Information

Eidesstattliche Erklärung

Hiermit versichere ich, dass diese Abschlussarbeit von mir persönlich verfasst ist und dass ich keinerlei fremde Hilfe in Anspruch genommen habe. Ebenso versichere ich, dass diese Arbeit oder Teile daraus weder von mir selbst noch von anderen als Leistungsnachweise andernorts eingereicht wurden. Wörtliche oder sinngemäße Übernahmen aus anderen Schriften und Veröffentlichungen in gedruckter oder elektronischer Form sind gekennzeichnet. Sämtliche Sekundärliteratur und sonstige Quellen sind nachgewiesen und in der Bibliographie aufgeführt. Das Gleiche gilt für graphische Darstellungen und Bilder sowie für alle Internet-Quellen.
Ich bin ferner damit einverstanden, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs in elektronischer Form anonymisiert versendet und gespeichert werden kann.

Datum	Marcel Mildenberger